

The Geoscience Laser Altimeter System (GLAS) I-SIPS Project

Integration Test Plan

1. Introduction

The process of software unit integration involves preparing a Build Plan consisting of several Builds, developing software integration test plans and test procedures for each Build, aggregating unit tested low level software units (SUs) into progressively larger SUs, testing them, and preparing software integration test reports. System testing is not included.

Purpose

This plan defines the procedure that the software integrator or Tester will use to perform integration testing for the GLAS I-SIPS project. The objective is to verify the interfaces between the SUs and the proper operation of each required function based on design information and the I-SIPS software system requirements.

Scope

This procedure applies to all deliverable new, reengineered or modified existing code for GLAS. All code to be integrated must have been unit tested according to the procedures set out in the Unit Test Plan document. Wherever possible, the Integration test may draw from the unit test cases.

Integration builds will be linked to milestones and will be scheduled. Integration test plans will be defined from detailed design functionalities and will be in accordance with the requirements of the software version to be delivered. Integration test procedures are derived during coding and unit testing. The test results will be reported during integration and testing.

Revisions and waivers

Revisions and waivers to this procedure must be authorized by the Change Control Board.

Definitions and Terms

Build – an interim product of SU integration. It should not be confused with the completed software, designated for phased deliveries as Version 0 (V0), Version 1 and Version 2.

Build Plan – a document that identifies builds and their contents and describes dependencies among builds and established integration schedules.

Integration Test Procedure – the set of steps that are performed in the testing of one or more functions or group of units defined for a Build. It includes all test cases and is based on the detailed design and the software requirements.

SU Integration - the aggregating of successfully unit tested SUs into threads, then aggregating threads into builds (functions of the subsystem) and builds into the complete subsystem. The steps in the SU integration follow the build plan. Integration tests verify that the integrated software implements the documented design and satisfies the stated requirements.

Thread - the necessary units to support the verification of a software function or capability from its input source to its output.

CSCI/SU- these terms refer to Computer Software Configuration Item and Software Unit. For the purpose of Configuration Management, they may be translated to Computer Program Component/ Configuration Item (CPC/CI) as described in the CM plan. In Clearcase the SUs are called software elements.

Test Conditions

Testing will incorporate units and libraries already unit-tested and placed under the Clearcase configuration control as “software elements” on the integration branch. Testing is completed when all functionalities defined for the build have been verified. For Version 0, testing will be done mainly in a top down fashion. The Executor(GLAS_Exec) will be the parent module and driver in all integration builds and tests when this approach is used.

Reference Documents

This test plan is in accordance with the GLAS Science Software Management Plan Section 8.1, 8.2 “The Assurance Plan”

2. Procedure

Although this procedure is described as a logical sequence of steps and tasks, SU integration and testing may require frequent replanning to overcome issues such as equipment failures, resource limitations, slipping schedules, and unexpected problems. The day to day activities will need to be monitored.

Developing the Build Plan

Consider the following when creating the build plan:

- a. functional dependencies
- b. control and data flows in preliminary design documents
- c. early integration of high-risk software
- d. availability of test tools and drivers
- e. software releases needed for system testing
- f. availability of hardware and unit tested software
- g. build size and complexity
- h. project schedule

A combination of a top-down and a bottom up implementation will be used in the build plan. In the top down approach, threads and builds will be the principal drivers of SU integration. The plan establishes the threads and the relationships between threads. The threads are used to integrate and test SUs and produce the subsystem, also referred as the Computer Software Configuration Item(CSCI) having increasing capability and maturity. One or more of these threads support a functional capability as defined in the software requirements specification. The build plan identifies these as milestones.

A milestone may consist of one or more capabilities that can be demonstrated by executing the SU integration test procedures. Major milestones in the integration and test schedule must support the program schedule. The critical path for major milestones should be identified in the integration and test schedule to allow for closer monitoring and faster problem resolution of critical activities

Depending on hardware, software, and human resource availability, parallel integration and testing may be scheduled to promote integration and testing efficiency. Integration testing will be done independently by either a designated tester, or members of the Software development Team(SDT) who have not developed the particular code to be tested.

Intermediate integration and test builds may be identified which consist of informal builds to allow for smaller increments of software to be integrated and tested. Smaller increments are encouraged, because in general, they promote faster problem identification.

Perform the following activities:

- a. Identify a nucleus set of SUs to meet basic functional requirements.
- b. Identify SUs that, when added to the nucleus, incrementally satisfy remaining software requirements.
- c. Provide for an early start on critical and/or complex functions.
- d. Use parallel paths where possible.
- e. Identify other software and hardware dependencies.
- f. Support system integration needs and focus on requirements for the scheduled delivery.
- g. Predict development of the software while supporting project and customer needs

Draw a Build Plan Dependency chart showing all the planned Builds. A build plan data sheet should be completed for each bubble(Build) in the dependency chart. Each bubble represents an increment in the process of integration and the informal testing of new or modified code.(See figure 1).

Build Plan Data Sheet

Build Number:	
Increment Identifier: (Build Name)	
Title:	
Specification: (Requirement)	
Description:	
Exit Criteria:	
Start date:	
End date:	

1. Document the build plan. (See format in Appendix E.)
2. Plan and schedule an internal review of the plan. Use a checklist to evaluate. (See appendix B) Track items to closure. Place documentation in the UDF. (Note that Appendix A defines the defect codes).
3. Use the build plan to determine the order in which units should be designed, coded, and tested.

Prepare integration test plan and procedures.

1. Plan to test each Build. Identify and document test cases, with objectives, functions being tested, relevant assumptions and the test environment. Give specific inputs, expected results and evaluation criteria. It is desirable that tests include a measure of actual resource utilization. See Appendix F section a) and b) for format.
2. Develop test procedures for each function or group of units being tested. Be detailed enough that it may be reproduced. Give details of the setup, including drivers, stubs, utilities, and data files. See the format in Appendix F section b).
3. Review the plan and procedures(Use the checklists in Appendix B and C for reviewing the plan and procedures).

Verify test Readiness:

Any material needed during SU integration should be readily available.

A version of the software should be ready for testing.

There should exist documentation on how to load and start up the system.

Source listings should be available.

Perform Integration and testing

Integrate according to the build plan. Document the software used.

Conduct Regression tests as appropriate for selected incremental builds. Regression testing, also known as benchmarking, will utilize a subset of the integration tests previously used to verify the software capability.

Execute procedures and record results.

Prepare an integration test report. See the format in Appendix F section c). and use the checklist in Appendix D to validate the report.

Conduct an internal review. Review test cases, results. Track action items to closure.

Place the test documentation in the UDF.

Appendix A

Defect Classification

I1n Incorrect Procedure

- I10 Other. Describe defect.
- I11 Logic error in procedure
- I12 Misunderstanding of design or requirements(e.g. procedure verifies the wrong thing)
- I14 Capability or accuracy beyond requirements expected.
- I15 Setup of software or hardware configuration is incorrect
- I16 Procedure is too intrusive (e.g. program sequence or test data affected)
- I17 Inefficient, redundant or unreasonable testing
- I18 Not compliant with parent document (e.g. test plan) or other documents.

I2n Incomplete Procedure

- I20 Other. Describe defect.
- I21 Insufficient test coverage
- I22 Test Environment not specified
- I23 Test objectives not stated
- I24 Expected results missing
- I25 Test setup (e.g. of initial data) or driver incomplete.
- I26 Insufficient description of responsibilities and resources
- I27 Insufficient description of limitations and constraints.

I4n Document deficiency

- I40 Other – describe the defect
- I41 Applicable document standards/format not met
- I42 Needs additional clarification/comments(e.g. to describe test results)
- I43 Spelling grammar, typographical errors
- I44 Document not current.
- I45 Inconsistent with other parts of document

I9n Enhancement

- I90 Other – Describe
- I91 Prerequisite data, test tool, or baseline software/hardware not available
- I92 Simplify

- I50 test procedure not followed

Appendix B

Build Plan Review checklist

Correct and complete	Defect Type
All functional requirements are addressed	I21
All functional dependencies are addressed	I21, I22
High risk software is integrated early	I15, I17
Test tools and drivers are identified	I22, I25
The build plan includes a schedule of start and finish dates	I15
The build plan will support testing activities that depend on it	I14
There are no logic errors in the build plan	I11
Consistent	
The order in which units are to be included in builds and tested is consistent with the project's development schedule	I15, I18
Feasible	
Hardware required for testing will be available	I15, I91
Test tools and drivers will be available when needed	I91
Each incremental build is of manageable size and complexity	I92
Meets Standards	
The build plan contains the information required by the document standard	I42
The build plan meets the format requirements of the document	I41

Appendix C

Integration Test Procedure Review Checklist

Correct and Complete	Defect Type
Interfaces between software and hardware are verified	I10, I21
Interfaces between units are verified	I10, I21
Expected output and response times are documented	I24
Software design requirements allocated to the builds are verified.	I10, I12, I21
Error conditions are verified (i/o errors, computational errors, processing overload, buffer overflow, events failing to occur.	I1,I21
Constants are verified	I10, I21
Resources (time, memory) are verified.	I10, I21
Test procedures meet the objectives established by the test cases	I18,I21
Consistent	
Test procedures are consistent with the test cases	I18
Test procedures are internally consistent	I45
Feasible	
Test conditions are reproducible	I15
Meets standards	
Test procedures contain the information required by the document standard	I41
Test procedures are developed in accordance with the software development plan	I18, I41
Test procedures adhere to the required project format	I41

Appendix D

Integration Test report Review Checklist

Correct and Complete	Defect Type
The test report describes the results of conducting the test procedures completely and accurately.	I24, I42
The test report clearly indicates success or failure.	I23, I42
The test report lists any action items resulting from the conduct of the test.	I26, I42
Consistent	
The test reports are consistent with the test procedures	I18
The reports are internally consistent	I45
Meets standards	
The test reports contains the information required by the product standard	I41
The test reports are developed in accordance with the Software Development Plan	I18, I41
The test reports adhere to the required project format	I41

Appendix E

Build Plan Format

Title:

Purpose:

Thread definitions:

Requirements:

Build Dependencies:

Resources required:

Databases needed:

Exit criteria:

Notes:

Appendix F

a) Integration Test Plan Format

Build Number:

Date:

Purpose:

Test Environment/Assumptions:

Test setup:

Functions used:

Test Execution:

b) Test Procedure Format:

Date:

Execution Steps	Inputs	Expected results	Observed Results	Pass/Fail	Comments	Resource Utilization

Test case 1:

Procedure:

Results:

Test Case 2:

Procedure:

Results:

c) Integration Test report

Summary Page:

Date:

Test Name/ID:

Version of software:

Version of Hardware and support software:

Test procedure date:

Engineer:

Problem report Number:

Defect detected during test conduct:

Action Taken:

Defect detected during test conduct:

Action Taken:

Is software ready for next activity?

Tester's Signature:

Test Procedure:

Build Dependency Chart

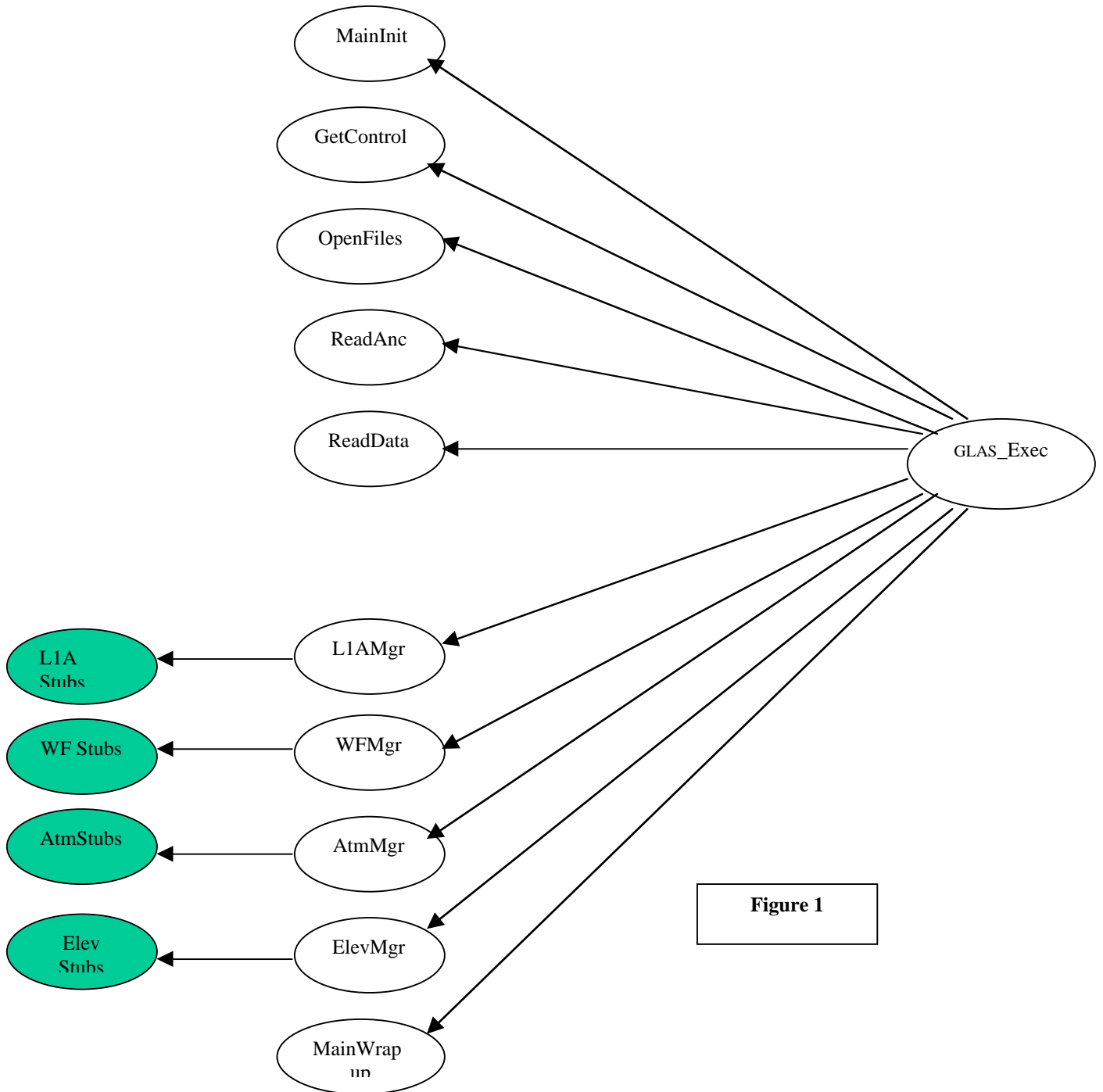


Figure 1